OLCF-6 LAMMPS Benchmark

Authors: Nick Hagerty, Van Ngo, and Dilip Asthagiri

This repository describes the Materials Modeling benchmark using LAMMPS. The OLCF-6 benchmark run rules should be reviewed before running this benchmark.

This benchmark derives from the Materials by Design Workflow NERSC-10 benchmark (https://www.nersc.gov/systems/nersc-10/benchmarks/). However, there are major differences in the type of systems considered. Also, there are only two categories of performance optimization, "ported" and "optimized." This is in contrast to the three categories in the Materials by Design Workflow.

Note:

- The OLCF-6 Benchmark Run Rules apply to this benchmark except where explicitly noted within this README.
- The run rules herein define the "ported" category of performance optimization. Responses to the OLCF-6 RFP should include performance estimates for this category; results for "optimized" categories are optional.
- RFP responses should estimate the performance for the target problem on the target architecture. The projected walltime for the target problem on the target system must not exceed the reference time measured by runnning the reference problem on Summit.
- Concurrency adjustments (i.e. weak- or strong-scaling) may be needed to match the reference time.
- The "capability factor" (c) descibes the increase in computational work (e.g. flops) between the reference and target problems, and may be used to guide resource requirements for the target problem.

Materials Simulation Overview

Molecular dynamics (MD) simulations help one understands the properties of matter from the knowledge of the inter-molecular interactions between the atoms/molecules/particles comprising the system. MD has revolutionized our understanding of materials in domains as diverse as biology, chemistry, physics, and material science. Modeling materials requires being able to describe both short-range and long-range interactions, each of which stresses different aspects of the computational approach, and thus different aspects of the computing architechture itself. To this end, the Materials Modeling benchmark uses two sub-problems, one that stresses the balance of short- and long-range interactions, and another that stresses short-range interactions involving changes that occur at a fast time-scale.

Liquid water is the solvent matrix for all biological processes, and to understand how biomolecules function, it is essential to understand the properties of water itself. A proper description of the thermodynamics of water depends on the correct description of the balance of long-range and short-range forces. This is in constrast to other so-called van der Waals fluids that can be well described by simply accounting for short-range interactions. Thus, the first of the two benchmarks comprising the OLCF-6 Materials benchmark involves the modeling of liquid water at room temperature.

Biology also relies on chemical transformations, and chemical transformations involve bond making/breaking. The most rigorous description of chemical transformations require electronic structure calculations. However, much progress can be made by using an empirical description of reactivity. One such approach is the so-called ReaxFF approach. In ReaxFF, one identifies the distribution of atoms around a given atom, assesses the bond order, iteratively equilibrates the charge distribution, and uses this information to drive the next step in the dynamics. Clearly, the ReaxFF approach stresses the local (short-range) interactions. Thus, the second of the two benchmarks comprising the OLCF-6 Materials benchmark involves the ReaxFF modeling of Pentaerythritol tetranitrate (PETN), an explosive material and a good test case for modeling reactive transformations at short time scales.

Code access and compilation

The process of building the benchmark has three basic steps: obtaining the source code, configuring the build system, and compiling the source code. The build instructions in this sections follow the build_lammps_cpu.sh script, which is suitable for a generic Linux workstation without a GPU accelerator. Detailed instructions for compiling the code on OLCF's Summit supercomputer using just CPUs and with GPU acceleration can be found in the build directory of this repository.

Obtaining LAMMPS source code

The required LAMMPS version used for this benchmark, update 4 of version 23 June 2022, is provided with this benchmark distribution. For reference, it was obtained using the commands below. Optimized runs may use custom modifications of LAMMPS.

```
git clone https://github.com/lammps/lammps.git lammps_src
cd lammps_src/
git checkout stable_23Jun2022_update4
```

Kokkos version 3.6.01 is distributed with and used by this LAMMPS version. Optimized results may use later versions of Kokkos or custom Kokkos backends optimized for the target architecture, as well as code modifications that are expressed in languages or abstractions other than Kokkos.

Configuring the LAMMPS build system

LAMMPS uses the CMake tool to configure the build system and generate the makefiles. From within the lammps_src directory, run the CMake commands that is most appropriate for your compute architecture. The example below is suitable for generic Linux workstation without a GPU accelerator. Additional cmake options that may be useful when customizing for other systems/architectures can be found in chapter 3 of the LAMMPS User Guide.

```
cmake -D CMAKE_INSTALL_PREFIX=$PWD/../install_cpu/ \
      -D CMAKE_CXX_COMPILER=g++ \
      -D CMAKE_Fortran_COMPILER=gfortran \
      -D BUILD_MPI=yes \
      -D MPI_CXX_COMPILER=mpicxx \
      -D PKG_USER-OMP=ON \
      -D PKG_KOKKOS=ON \
      -D DOWNLOAD_KOKKOS=ON \
      -D Kokkos_ARCH_FIXME=ON \
      -D PKG_ML-SNAP=ON \
      -D CMAKE_POSITION_INDEPENDENT_CODE=ON \
      -D CMAKE_EXE_FLAGS="-dynamic" \
      ../cmake
```

LAMMPS configuration used in Summit benchmark

LAMMPS Version

This benchmark was constructed using the LAMMPS commit `59e8b9370f`, which at the time of writing corresponds to the tag `patch_23Jun2022_update4`. See https://docs.lammps.org/Build_make.html (https://docs.lammps.org/Build_make.html) for more information about building LAMMPS with the Make-based build system.

Required Packages

The supplied LAMMPS benchmarks require the following packages:

- KSPACE -- for PPPM
- MOLECULE -- for `atom_style full`
- REAXFF -- for ReaxFF potential
- RIGID -- for `fix shake`

And additional packages for acceleration:

- KOKKOS -- for CPU (OpenMP) and GPU acceleration or
- OPENMP -- for CPU-based acceleration

Software Prerequisites

- A C++14 capable compiler
- GNU make
- Optional: hwloc
- Optional: LAMMPS-supported FFT library for KSPACE (ie, FFTW3)

How to Build:

There are 2 build scripts provided:

- build_lammps_rfp.sh -- builds an OpenMP-accelerated binary of LAMMPS using FFTW3 for FFTs.
- build_lammps_gpu.sh -- builds a GPU-accelerated binary of LAMMPS using Kokkos with both CUDA and OpenMP backends, and FFTW3 for FFTs.

build_lammps_rfp.sh

This script sets up the environment used on the Summit POWER9 supercomputer at Oak Ridge National Laboratory (ORNL) for CPU-only runs. The Makefile, which is located in `makefiles/Makefile.summit_cpu`, uses the GCC `g++` compiler & linker. The build script copies this script into `lammps/src/MAKE/MINE/Makefile.summit_cpu`, where it is found by the LAMMPS build system. To use the internal LAMMPS KISS-FFT instead of FFTW3, remove the define, include, and link lines in the `Makefile.summit_cpu`.

build_lammps_gpu.sh

This script sets up the environment used on the Summit POWER9 supercomputer at Oak Ridge National Laboratory (ORNL), with NVIDIA V100 support. The Makefile, which is located in `makefiles/Makefile.summit_gpu`, uses the Kokkos `nvcc_wrapper` for compilation and linking. The build script copies this script into `lammps/src/MAKE/MINE/Makefile.summit_gpu`, where it is found by the LAMMPS build system. To use the internal LAMMPS KISS-FFT instead of FFTW, remove the define, include, and link lines in the `Makefile.summit_gpu`. Kokkos is compiled with OpenMP and CUDA support. The PPPM solver utilizes the OpenMP backend of Kokkos due to poor scaling with the CUDA backend.

Running the benchmark

Input files and batch scipts for the various systems are provided in the benchmarks directory. Each problem has its own subdirectory within the benchmarks directory. There the appropriate run scripts are included.

Responses to the OLCF-6 request for proposal (RFP) should provide results (measured or projected) for the "target" problem size. Figure-of-Merit (FOM) values from OLCF Summit system were evaluated using the "reference" problem size. Other problem sizes have been provided as a convenience, to facilitate profiling at different scales (e.g. socket, node, blade, or rack), and extrapolation to larger sizes. This collection of problems form a weak scaling series where each successively larger problem simulates eight times as many atoms as the previous one. Computational requirements are expected to scale linearly with the number of atoms.

The following table lists the approximate system sizes of each of these jobs. The simulation log files generated on Summit provide additional details such as memory and number of CPUs/GPUs used (see also the Results section below). The capability factor (c) parameter is an estimate of the computational complexity of the problem relative to the "reference" problem.

SPC/E water simulations

Index	Size	Replication factor	# atoms	Capability factor
0	Tiny	48 x 48 x 48	5.308 M	$1/8^3$
1	Small	96 x 96 x 96	42.467 M	$1/8^2$
2	Medium	192 x 192 x 192	339.738 M	$1/8^1$
3	Reference	384 x 384 x 384	2.7179 B	1
4	Target	768 x 768 x 768	21.743 B	8

PETN ReaxFF simulations

Index	Size	Replication factor	# atoms	Capability factor
0	Nano	35 x 35 x 35	2.487 M	$1/8^3$
1	Tiny	70 x 70 x 70	19.894 M	$1/8^2$
2	Small	140 x 140 x 140	159.152 M	$1/8^1$
3	Reference	280 x 280 x 280	1.273 B	1
4	Target	560 x 560 x 560	10.186 B	8

Parallel decomposition

LAMMPS uses a 3-D spatial domain decomposition to distribute atoms among MPI processes. The default decomposition divides the simulated space into rectangular bricks. The proportions of the bricks are automatically computed to minimize surface-to-volume ratio of the bricks. LAMMPS will run correctly with any number of MPI processes, but better performance is often obtained when the number of MPI processes is the product of three near-equal integers. If additional control of the domain decomposition is needed, the processors command may (optionally) be added to the input files. This command requires three integer parameters that correspond to the x,y,z dimensions of the process grid. Changes to processors may require updates to the job submission script so that the correct number of MPI processes is started. Refer to the LAMMPS documentation for more information about the processors command. The parameters of the LAMMPS input file may not be modified except for the addition of a processors command.

Job submission

The essential steps are:

1. add symlinks to the files that are common to all runs. These files are the LAMMPS configuration file `input.in` and the data/forcefield descriptions that are common to the SPC/E water simulations and ReaxFF simulations, respectively. These common files are in `benchmark-main`.
2. the size-specific simulation parameters are in the file `size_cmd.txt`. (See also Table above.) This file contains the information on the number of times the simulation cell is replicated in x, y, and z directions.
3. run the job. For the CPU-only runs, please refer to `run_cpu.bsub` and for the GPU accelerated runs please refer to `run_gpu.bsub`.

Results

For both the SPC/E water and ReaxFF simulations, the results obtained with GPU-acceleration set the reference results that must be met or exceeded. For the SPC/E water simulations, we additionally provide CPU-only results to validate the results obtained using GPU-acceleration. The `validate.py` script can be used for this task. For the ReaxFF simulations, we only provide results using GPU-acceleration. The figure out merit (FOM) is `atom-steps/sec`. The `validate.py` script also prints the FOM.

The `# Nodes` below indicates the number of Summit nodes that were used. The job submission script(s) and/or the log files give a finer breakdown of the number of cores and GPUs used.

Reference performance on Summit

SPC/E water simulations

Index	Size	# Atoms	# Nodes	Capability factor	FOM (atom-steps/sec/10^6)
0	Tiny	5.308 M	1	$1/8^3$	27.7
1	Small	42.467 M	8	$1/8^2$	94.7
2	Medium	339.738 M	64	$1/8^1$	146.7
3	Reference	2.7179 B	512	1	1872.2
4	Target	21.743 B	4096	8	4957.7

Note on validation

For the SPC/E benchmark, the results from the CPU-only runs can be used as a reference to check the correctness of the results on the proposed systems, presumably using accelerators. As an example, consider the Tiny benchmark noted above. The `0_tiny` directory has two files, `lammps_benchmark_gpu.3120976` and `lammps_benchmark_cpu.3120975`. The first is the log produced using GPU acceleration, and second file is the log produced using a CPU only (the reference). These files become inputs to the `validate.py` script under `benchmarks`. The example output looks as follows.

```
python ../../validate.py --input lammps_benchmark_gpu.3120976 --rfp lammps_benchmark_cpu.3120975

Checking: lammps_benchmark_gpu.3120976 against lammps_benchmark_cpu.3120975 for correctness.

Checking energy breakdown. Ignoring the following fields: Pxx, Pyy, Pzz

LAMMPS run passed all validation with a FOM of 27664569.92469448 atom-steps/
```

Note that the `validate.py` script can take other options, such as including the components of the pressure tensor into the validation (turned off by default) or changing the threshold for agreement (default is 0.1). To user pressure tensor data, for example, in the development phase of the machine, please remember to turn on output of pressure tensor information during the LAMMPS runs (turned off by default in the scripts provided with this benchmark.

PETN ReaxFF simulations

Index	Size	# atoms	# Nodes	Capability factor	FOM (atom-steps/sec/10^6)
0	Nano	2.487 M	1	$1/8^3$	15.2
1	Tiny	19.894 M	8	$1/8^2$	110.9
2	Small	159.152 M	64	$1/8^1$	886.1
3	Reference	1.273 B	512	1	6564.7
4	Target	10.186 B	4096	8	46442.5

Reporting

Benchmark results should include projections of the FOM for the target problem size. The complete hardware configuration needed to achieve the estimated timings must also be provided. For example, if the target system includes more than one type of compute node, then report the number and type of nodes used to run the benchmark. For the electronic submission, include all the source and makefiles used to build on the target platform and input files and runscripts. Include all standard output files.